# **Exercícios de Fixação**

# **(3) Estruturas de Programação**

**1)** Dado os comandos abaixo, marque a alternativa **CORRETA** quando executarmos todos esses comandos juntos:

SELECT 'ANTES DA DIVISÃO POR ZERO'

SELECT 2 / 0

SELECT 'DEPOIS DA DIVISÃO POR ZERO'

a) Como temos uma divisão por zero, nenhum dos comandos será executado.

b) Como temos uma divisão por zero, isso irá gerar um erro e nenhum dos scripts será executado.

c) Apesar da divisão por zero, os outros dois SELECTs serão executados normalmente.

d) Sei não, mas é alguma das respostas acima aí Luiz.

**2)** Dado os comandos abaixo, marque a alternativa **CORRETA** quando executarmos todos esses comandos juntos:

SELECT 'ANTES DA DIVISÃO POR ZERO'

SELEC 2 / 0

SELECT 'DEPOIS DA DIVISÃO POR ZERO'

a) Nesse caso, temos 3 batches distintos.

b) Como temos um erro de sintaxe no segundo SELECT e apenas um batch, nenhum comando será executado.

c) Apesar da divisão por zero e do erro de sintaxe do segundo SELECT, os outros dois SELECTs serão executados normalmente.

d) Vou precisar executar os comandos aqui pra conseguir responder Luiz.

**3)** Dado os comandos abaixo, marque a alternativa **CORRETA**:

SELECT 'ANTES DA DIVISÃO POR ZERO'

GO

SELEC 2 / 0

GO

SELECT 'DEPOIS DA DIVISÃO POR ZERO'

GO

a) O comando GO é uma instrução T-SQL utilizada para sinalizar o término de um batch. Ele é reconhecido em qualquer utilitário de banco de dados, independentemente de ser SQL Server ou não.

b) Nesse caso, temos apenas 1 batch.

c) Como temos um erro de sintaxe no segundo SELECT, nenhum dos três comandos será executado.

d) Temos um erro de sintaxe no segundo SELECT, mas temos 3 batches distintos. Com isso, apenas o segundo SELECT irá falhar e os outros dois serão executados normalmente.

**4)** Dado os comandos abaixo, marque a alternativa **CORRETA**:

SELECT 1

SELECT 2

GO 2

a) Irá gerar um ERRO, pois o comando não é válido.

b) O resultado será: “1”, “2”, “2”.

c) O resultado será: “1”, “2”, “2”, “2”.

d) O resultado será: “1”, “2”, “1”, “2”

**5)** Dado os comandos abaixo, marque a alternativa **CORRETA**:

(1) WAITFOR DELAY '00:00:05'

(2) WAITFOR TIME '16:20'

a) O comando (1) irá esperar por 5 minutos.

b) O comando (2) irá esperar por 16 minutos e 20 segundos.

c) O comando (2) irá esperar até às 16:20 horas para terminar a execução.

d) O comando (1) irá esperar até meia-noite e cinco minutos

**6)** Assinale a alternativa **CORRETA**:

(1) Os comandos PRINT, SELECT e RAISERROR funcionam da mesma forma e retornam os mesmos resultados.

(2) Os comandos PRINT e SELECT funcionam da mesma forma e retornam os mesmos resultados.

(3) Apesar do nome, o comando RAISERROR não gera um erro na execução da query.

(4) No comando abaixo, o SELECT não será executado.

RAISERROR ('Usando o RAISERROR para gerar um ERRO!', -- Message text.

16, -- Severity.

1 -- State.

);

SELECT 'ESSE SELECT É EXECUTADO DEPOIS DO RAISERROR!'

a) F, F, F, F

b) F, F, V, V

c) F, F, F, V

d) F, V, V, V

e) V, V, V, V

**7)** Assinale a alternativa **CORRETA**:

(1) Ao usar um IF, o comando ELSE é opcional.

(2) O comando abaixo irá retornar o valor “1”.

DECLARE @VALOR INT = 1

IF (@VALOR)

SELECT 1

GO

(3) Ao usar um IF, é obrigatório a utilização do BEGIN ... END para delimitar o escopo dos comandos.

(4) Um IF não pode ter outros IFs aninhados.

a) V, F, F, F

b) V, F, V, F

c) V, F, V, V

d) F, F, V, V

e) F, V, F, F

**8)** Dado os comandos abaixo, assinale a alternativa **CORRETA**:

-- COMANDO 1:

DECLARE @VALOR INT = 5

SELECT IIF(@VALOR < 10, 'O VALOR É MENOR DO QUE 10!', 'O VALOR É MAIOR DO QUE 10!') AS Result;

-- COMANDO 2:

SELECT CHOOSE (3, 'Manager', 'Director', 'Developer', 'Tester') AS Result;

a) Os comandos IIF e CHOOSE utilizam a mesma ideia. Portanto, são comandos semelhantes.

b) O comando CHOOSE irá repetir os parâmetros por 3 vezes no resultado.

c) O comando CHOOSE valida uma condição e se for verdadeira executa um comando, caso contrário executa o outro.

d) O comando IIF valida uma condição e se for verdadeira executa um comando, caso contrário executa o outro.

**9)** Dado o comando abaixo, o valor retornado será:

DECLARE @TESTE\_CASE INT = 15

SELECT

CASE

WHEN (@TESTE\_CASE - 5) = 10

THEN 'O RESULTADO É 10'

ELSE

'O RESULTADO É DIFERENTE DE 10'

END AS Teste\_Case

a) "O RESULTADO É 10", pois o valor da variável é 15, logo é diferente de 10.

b) "O RESULTADO É 10", pois o resultado da expressão no WHEN será verdadeiro.

c) "O RESULTADO É DIFERENTE DE 10", pois o resultado da expressão no WHEN será falso.

d) "O RESULTADO É DIFERENTE DE 10", pois o valor da variável é 15, logo é diferente de 10.

e) ERRO, pois o comando está com um erro de sintaxe no uso do CASE.

**10)** Dado o comando abaixo, o valor retornado será:

DECLARE @LOOP INT = 1

WHILE(@LOOP <= 10)

BEGIN

IF ((@LOOP % 2) = 0)

BEGIN

SET @LOOP = @LOOP + 1

CONTINUE

END

PRINT @LOOP

SET @LOOP = @LOOP + 1

END

GO

a) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

b) 1, 2, 3, 4, 5, 6, 7, 8, 9

c) 2, 4, 6, 8, 10

d) 2, 4, 6, 8

e) 1, 3, 5, 7, 9

**11)** Marque a alternativa **CORRETA** sobre Stored Procedures:

(1) Não podemos fazer tratamentos de erro dentro das Stored Procedures.

(2) Não podemos fazer alterações no banco de dados dentro das Stored Procedures.

(3) Ajuda no controle de segurança, mas não previne o SQL Injection.

(4) Uma vez que o plano de execução de uma Stored Procedure for criado, ele sempre terá uma boa performance independente dos parâmetros que forem utilizados.

a) F, F, F, F

b) F, F, F, V

c) F, V, F, F

d) F, F, V, V

e) V, F, F, V

**12)** Dado o script abaixo, qual será o resultado?

CREATE PROCEDURE [dbo].[stpRetornaDobro] (

@Vl\_Entrada INT NULL = 0 OUTPUT

)

AS

BEGIN

SET @Vl\_Entrada = @Vl\_Entrada \* 2

END

GO

-- RESULTADO

DECLARE @VALOR INT = 10

EXEC [dbo].[stpRetornaDobro] @VALOR

SELECT @VALOR

a) 10

b) 20

c) 0

d) 2

e) Vish, to na dúvida Luiz, explica ai por favor.

**13)** Dado o script abaixo, qual será o resultado?

CREATE PROCEDURE [dbo].[stpParametroDefault] (

@Val1 INT = 100,

@val2 VARCHAR(100) = 'Teste Parâmetro Default'

)

AS

BEGIN

SELECT @Val1 AS Val1, @val2 AS val2

END

GO

-- RESULTADO

EXEC [dbo].[stpParametroDefault] 'Luiz Vitor'

a) ‘Luiz Vitor’, ‘Luiz Vitor’

b) Vai dar um erro de conversão no primeiro parâmetro.

c) ‘Luiz Vitor’, ‘Teste Parâmetro Default’

d) 100, ‘Luiz Vitor’

e) Pera aí que vou executar ela aqui e já te respondo Luiz.

**14)** Marque a alternativa **CORRETA** sobre functions:

(1) Não podemos fazer tratamentos de erro dentro das Functions.

(2) Não pode fazer alterações no banco de dados dentro das Functions.

(3) Existe apenas um tipo que é a Scalar Function, que retorna apenas um valor.

(4) Utilizar Functions é uma boa prática e não tem problemas relacionados a performance.

a) F, V, F, F

b) F, V, V, F

c) V, V, F, F

d) V, F, F, V

e) V, V, F, V

**15)** Marque a alternativa **CORRETA** sobre views:

(1) É uma boa prática utilizar o SELECT \* em uma VIEW, pois dessa forma já irá retornar todas as colunas da tabela sempre, mesmo quando a tabela for alterada (por exemplo, adicionar uma nova coluna).

CREATE VIEW [dbo].[vwClientesRicos]

AS

**SELECT \***

FROM [Cliente]

WHERE Vl\_Salario >= 30000.00

GO

(2) Pode ajudar a reduzir o código, pois pode simplificar um SELECT grande que envolve vários filtros ou tabelas.

(3) Maior controle de segurança, pois pode liberar o acesso apenas na VIEW e não nas tabelas diretamente.

a) V, V, V

b) V, V, F

c) F, V, F

d) F, V, V